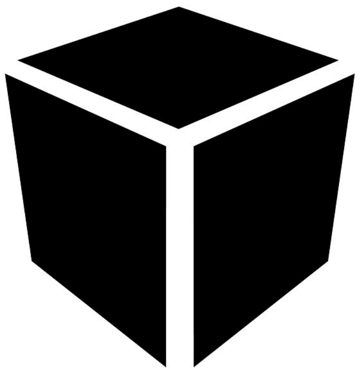
How do you explain that?
An assessment of black box model explainers

Marnix Maas

January 28, 2020



Contents

Abstract

Recent developments in machine learning have created methods to provide ~~a~~ human interpretable explanations for any predictive model. This research compares model-agnostic explainers Lime and Shap in three different simulated experiments. -It is suggested that Shap provides the best explanations for most models. However, the Lime explainer shows a marginally lower performance, while requiring less computation time. Secondly, a novel explainer evaluation metric is tested, and deemed inadequate in its current implementation.-

Depends on results

1 Introduction

Decisions made by machine learning algorithms are often not interpretable by humans. Not even the most practised data scientists can expose exactly why their deep learning model chose the way it did. Still, with each day more predictive models are being implemented. Of which most are so hard to interpret, we tend to refer to these models as black boxes.
In light of this problem, advancements have been made to create methods that can provide an explanation for any model. Two of these so-called post hoc model-agnostic explainers are Lime and Shap. However complicated, these explainers intend to provide insight into all individual decisions of the algorithm.
The literature is divided; arguments are put forward that the use of such methods is flawed or not desirable, while others are making advancements into the explainers. I suggest that this division has occurred because the (non-)quality of explainers is hard to measure. Making things worse, the more simplified the explanation, the more likely it is to omit crucial information. If these problems can be surmounted the improved accuracy of black box models can be utilized, without sacrificing explanability.

This research aims to evaluate and compare two recently proposed model explanation methods. For this evaluation, two previously proposed simulated user experiments are carried out, with the addition of the latest explainer Shap. Several improvements to these existing experiments are proposed. Lastly, a novel experiment to evaluate explainers using generated data is proposed.
Secondly, this research examines a new metric for the evaluation of model explainers.

2 Literature review

Recent developments and research on post hoc model-agnostic explainers is firstly discussed. Thereafter, a novel method to assess the quality of such explainers is covered. And lastly, tendencies against the use of explanation methods are considered.
It has been suggested that the terms explanability and interpretability should not be used as synonyms, as they may be different in nuance[1]. However, in this work these terms will both be used to describe the explanation formed by one of the discussed methods. Waar plaatsen?

Post hoc model-agnostic explainers

In recent years, many advancements in linear post hoc model-agnostic explainers have been made. The quest for such methods seems to come from the trade-off between model performance and explanability. Most recently best performing machine learning algorithms, while having improved accuracy over previous methods, are hard or impossible to interpret. For example deep learning and ensemble methods. As a result, a new field in model explanability has arisen. Most versatile are those methods that can be applied to any model - model-agnostic - and after the model has been trained - post hoc -.
To this end, Ribeiro et al. [1] have introduced Lime in 2016. Lime being a Local Interpretable Model-agnostic Explainer. In short, the method creates a sparse linear explanation of an instance from the data, that is locally faithful. The explanation consists of a maximum of K features, a user specified number, which are assigned an attribution to the prediction. The explanation is additive, all feature attributions (and a base value) sum up to approximately the model's prediction for that instance. The explanations are locally faithful in the sense that the explanation should be correct in the vicinity of the instance, but is not guaranteed to be globally faithful. In addition, the paper argues that Lime fulfills above mentioned properties using some simulated experiments.
An interesting addition to the explanation model is the notion of coverage. Their SP-Lime algorithm will find a set of the most representative explanations, that set of explanations should optimally describe the global behaviour of the algorithm.

[1]Interpretability would suggest a degree to which the model's output can be predicted. While explanability covers the degree of being able to explain to a human the inner mechanics of an algorithm.

In a work by Lundberg and Lee [2] all additive feature attribution methods, like Lime, are joined under a single definition. They unite Lime, DeepLIFT, layer-wise relevance propagation and Classis Shapley value estimation. Then, it is shown that only Shapley values [10] can satisfy three desirable properties that enforce a unique solution for an additive feature attribution explanation. According to Shapley's theorem, that unique solution is optimal. Shortly, those properties describe local accuracy/faithfulness, no attribution if the feature is not included in the explanation and consistency of the feature attributions. Lastly, Lundberg and Lee propose a new method named Shap Kernel, to approximate Shapley values with improved sample efficiency. It should be noted that, Shap has brought the theoretical bases of Shapley values from game theory to additive feature attribution methods. However, it is assumed that features are independent and the explanation model is linear.
To counter these assumptions, an improved method was proposed by Aas et al. [3] in 2019. They suggest that the assumption of independency of features can produce faulty explanations. Thus, they propose an improvement of the Shap Kernel method to handle dependent features. Since the assumption of independency is only necessary in one step of calculating Shap values, they suggest relaxing that assumption. They found that for non-linear models the improved method outperforms Shap. A drawback of allowing dependencies is that explanations become harder to interpret. Multiple dependent features can only be properly interpreted as a group, rather than individually. Additionally, this extension to Shap increases computation time.
In addition to extending Shap for dependent features, an extensive overview of the Shap Kernel approximation method is given, since the original paper [2] does not fully describe the implementation. An accurate description is given of how the sample efficiency is improved over previous Shapley value approximations.

Aside from the creation of and improvements to model explainers, new methods have been proposed to evaluate them. A novel 'faithfulness' metric has been suggested by Arya et al. [5]. The method aims to evaluate the quality of an explanation using correlation. They have suggested a second evaluation metric named 'monotonicity', although that relies on similar evaluation of correlation. As of this date, no research has been done to evaluate their explainer faithfulness metric.

Disadvantages to post hoc explainers

The advancement in explanation methods is compelling, and shows potential for decreasing the trade-off between model interpretability and performance. However, some works have put forward that using post hoc explainers may not be beneficial.
In a recent paper from november 2019 by Slack et al. [7] a framework was created to fool Lime and Shap into producing faulty explanations. In short, the framework creates a clearly biased model and hides that bias by abusing the perturbations on which explainers rely. The explanations produced by the Lime and Shap seem unable to reproduce that bias. It should be noted that explainers are susceptible to "adversarial attacks" (2019: 6) as Slack et al. suggest. Thus, if one has wrong intentions, Lime and Shap can be fooled into faulty explanations.
In some domains, for specific models, post hoc explainers have been shown to miss the most relevant feature of an instance. One such example is from Camburu et al. [4] for explaining a neural network for a natural language processing task. Their goal is to create (a framework for) an evaluation test for post-hoc explanatory methods, from the perspective of feature-selection. Their evaluation only considers the inclusion of features, unlike the faithfulness measure from Arya et al.. While Camburu et al. conclude that post hoc explainers can miss the most relevant feature of a prediction, their framework was only tested on a single model for a single task. As they note in their conclusion, their framework could be applied to other tasks and areas.

3 Methods

This section will describe techniques used in the simulated user experiments. The post hoc model explainers Lime and Shap are discussed in detail. In addition, an overview of the global explanation method 'parzen windows' is explained. Parzen window is used as baseline comparison the the local explainers. Lastly, the method of data generation and a quantitative evaluation metric for explanations are considered.

3.1 Additive feature attribution methods

The explanation methods used in this paper fall into the category of 'additive feature attribution explanation methods', also called linear explainers. Linear comes from the fact that these explainers rely on sparse linear models for their explanations. The following sections will extend on the exact method.

Thought, it is useful to give an example of that additive feature attribution before going into detail. Consider figure 1 from Lundberg's repository[2].

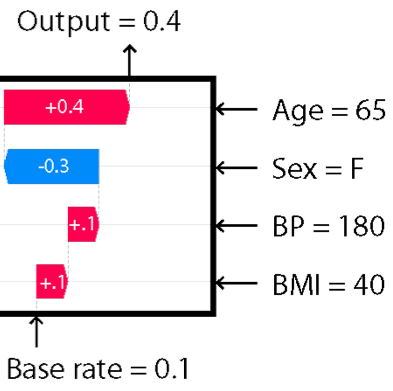


Figure 1: An example of additive feature attributions methods. Starting at the base rate (mean probability of that class) each feature contributes their attribution to end up at the output value. The positive attributions are shown in red, while negative attribution is shown in blue.

3.2 Perturbation

Post hoc model-agnostic explainers rely on feature perturbation in order to provide explanations while treating the classifier as a black box. To measure the influence of a single feature on the prediction, it is taken out of the instance, i.e. the initial value. Most machine learning methods do not allow a feature to have no value. Thus, the value is replaced, in order to mask its impact. The value can be replaced by the most common/average value from the background dataset or a value of 0. A sensible choice for that replacement depends on the data. The process of masking a feature from an instance is called perturbation. An advantage is that perturbation is possible for any classification model.
Perturbation or the masking of features can also be done for multiple features of an instance. For Lime and Shap a perturbed sample of an instance x is denoted as z or z'.

[2]https://github.com/slundberg/shap

3.3 Lime

The Lime (Local Interpretable Model-agnostic Explanations) method has been proposed by Ribeiro et al [1]. In short, Lime aims to provide a human interpretable explanation that is locally faithful to, and can be applied to any, machine learning model.
Firstly, interpretability, model-agnostic explainers and (local) faithfulness are characterized:

- An explanation is interpretable if it can be easily understood by humans. For Lime specifically, it assumes that a sparse linear model is interpretable. Sparsity meaning that the linear explanation incorporates a small number of features, that would allow for an easily readable explanation. E.g. a linear explanation with only 5 features should be quickly understandable for a human.
- The method is model-agnostic. The explanation method does not make any assumptions about the underlying predictive model. Therefore, it can be applied to any machine learning model.
- The explanations are locally faithful to the predictive model. According to Ribeiro et al. the explanation "must correspond to how the model behaves in the vicinity of the instance begin predicted" (2016: 3). Therefore, the explanation should be faithful to the model in a local manner, but it does not imply global faithfulness.

Formally, Lime has that g is an explanation with a limited amount of features included. The complexity of g for linear explanations is measured by the number of included features and is denoted as $\Omega(g)$. G is the family of interpretable models (for example, a small decision tree would also apply). Let $f(x)$ be the probability that x belongs to a certain class according to model *f*. The proximity (vicinity) between instance x and z is $\pi_x(z)$, where z is a perturbed sample of instance x. Then Lime is defined by:

$$\xi(x) = \underset{g \,\in\, G}{argmin}\ L(f, g, \pi_x) + \Omega(g)$$

Lime finds the explanation $\xi(x)$ that is the most locally faithful, while still being interpretable. A formal definition of L is included in appendix section **??**. The user can specify the complexity budget, or maximum allowed number of features K. When the explanation becomes larger than K, the complexity constraint $\Omega(g)$ will be infinite. This enforces the explanation to be smaller than K. The explanation will look as described in section 1. How the method works intuitively is shown in figure 2.

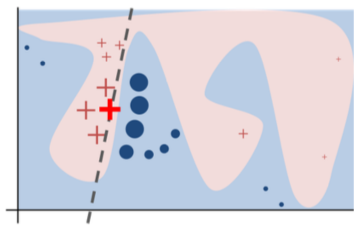


Figure 2: Source [1]. Intuition of Lime method: the linear explanation in dashed lines is constructed with weighted perturbed samples around the instance. The background color indicates the decision boundary of the model. This figure aims to show that an explanation can be locally, but not globally, faithful to the decision boundary of the model.

3.4 Shap

A unified approach to model explanation methods, named Shap, has been proposed by Lundberg and Lee [2]. In their work they aim to show that if a linear explanation is created for a model, Shap values are the most consistent and computationally viable.
Shapley values are a theorem from game theory, but they are used in the Shap explainer. Firstly, Shapley values are described. Thereafter, the use of Shapley values in the Shap explainer is considered.

Shapley values

Shapley values [10] are a method for distributing the total payout of several games over n persons, depending on their contribution to the payout for each game. According to the theorem, Shapley values are the optimal assignment of payout, since they adhere to a set of desirable properties. The following section will cover these properties in full, as they are distinctly for additive feature attribution methods. But, a formal definition of the properties is included in appendix section 7. Young (1985) has shown that a solution that adheres to those desirable properties is unique. Thus, according to these theorems Shapley values are the unique set of attribution for optimal distribution.

Molnar[3] (2019) accurately describes Shapley values: "A prediction can be explained by assuming that each feature value of the instance is a 'player' in a game where the prediction is the payout. Shapley values – a method from coalitional game theory – tells us how to fairly distribute the 'payout' among the features".

Unified definition - additive feature attribution methods

Shap Kernel is based on the Lime method, but with different choices for the weighting kernel π (distance metric) and regularization term Ω. For Lime, these parameters are chosen heuristically, whereas the Shap Kernel method defines these paramaters according to the Shapley theorem. Formally, in Lundberg and Lee's theorem 2, the Shapley kernel is defined as:

$$\Omega(g) = 0,$$

$$\pi_{x'}(z') = \frac{M-1}{(M\ choose\ |z'|)|z'|(M-|z'|)}$$

Where M is the amount of features, the amount of non-zero features in z' is $|z'|$ and x' is the set of all perturbations of the data. With this definition of the weight kernel and regularization, the estimated values should adhere to 3 desirable properties. Whereas the heuristic choice for Lime may result in a violation of local accuracy and consistency.

A formal definition of the desirable properties is given in appendix section 7. Intuitively, the properties can be explained as:

- Local Accuracy: The explanation exactly matches the predicted probability by the model for an individual. The mean explanation summed with all explained feature attributions are the same as the output of the model.

- Missingness: Features not included in the explanation do not have any attribution to the explanation.

- Consistency: Feature attribution should not decrease if the features' input is kept the same or increased, while the other inputs are unchanged.

[3]https://christophm.github.io/interpretable-ml-book/shapley.html, chapter 5.9

A unique additive feature explanation model follows from these properties:

$$\phi_i(f,x) = \sum_{z' \subseteq x'} \frac{|z'|!\,(M-|z'|-1)!}{M!}\,[f_x(z') - f_x(z' \setminus i)]$$

Where i indicates a specific feature.

Shap Kernel

The complexity of calculating Shapley values is dependent on the number of features. The exact calculation of Shapley values becomes computationally intractable for a large number of features. Since the exact calculation is computationally expensive, Lundberg and Lee have proposed Kernel Shap to approximate them. Under the assumption that features are independent and the model is linear, Shap values can be calculated with higher sample efficiency than previous Shapley equations. It should be noted that feature independence is an assumption when using perturbations, it is not specific to Shap values. The improved sample efficiency is described in detail in a different paper in section 2.3 by Aas et al. [3].

← Zou ik in moeten gaan op deze verbeterde efficientie? Zo ja, dat gaat naar appendix

In addition to the model-agnostic Shap Kernel method, Lundberg and Lee have introduced several model specific methods for calculating Shap values even more efficiently. These include Deep Shap for deep learning models and Tree Shap for tree based models. However, for this research only true model-agnostic methods are considered.

In summary, Shap introduces a faster method to approximate Shapley values. According to the theory, only Shapley values adhere to a triplet of desirable properties for an explainer. Like the Lime method, Shap will produce an explanation with feature attribution for up to K features.

3.5 Parzen

The Parzen-windows technique is an approach to estimate the probability density function of a specific point without knowing the underlying distribution. A region around the point is used to estimate the value of probability density. That region is where the name Parzen-windows comes from. Baehrens et al. [9] describe how Parzen-window can be used to explain individual classification decisions.
They define the Bayes classifier:

$$g^*(x) = arg \min_{c\in\{1,\ldots,C\}} P(Y \neq c | X = x)$$

Where C is the number of classes in the classification problem and $P(X, Y)$ is some unknown joint distribution. Then, the explanation vector of a data point x_0 is the derivative to x at $x = x_0$. Formally noted as:

$$\zeta(x_0) := \frac{\partial}{\partial x} P(Y \neq g^*(x) | X = x)\Big|_{x=x_0}$$

With $\zeta(x_0)$ a d-dimensional vector, with the same length as x_0. The explanation is formed by the largest (absolute) feature attributions in the vector $\zeta(x_0)$, up to K features. Note that Lime defined the explanation vector as $\xi(x)$.

3.6 Data generation

to be added
–generate data using sklearn make classification, mimic some stuff from other experiments. Also K=10 and do a binary classification.
–we can specify the number of informative features, noise, and whatever levels. We will play with these parameters to see sturdyness of explainers.

3.7 Faithfulness metric

The quality of an explainer depends on the interpretability offered to the user as well as the faithfulness to the model. These notions may have opposing effect. A simpler explanation may be less faithful to the model, as it cannot capture its full extend.
Current post hoc explanation methods use a (sparse) linear explanation. They should offer the same level of interpretability as long as they have the same amount of features. Thus, they could easily be compared based on some notion of faithfulness to the model.
Currently, there is no standard metric to measure that faithfulness. However, Arya et al. [5] have introduced a conveniently named faithfulness metric, called: 'faithfulness'. This metric aims to quantify the quality of explainers, rather than human evaluation being the golden standard. In this paper, the aim is to determine the quality of this metric.
The faithfulness metric expresses the quality of an explainer as correlation between model predictions and feature importance. In order of feature importance, the feature of an instance is replaced by the background value and the model's prediction probability is recorded. That process is repeated for all features, for which a feature importance exists. The faithfulness metric

ϕ is then defined as the negative pearson correlation ρ between the vector of feature importances Θ and the vector with the model's prediction probabilities $\mathbf{p}$:

$$\phi = -\rho(\Theta, \mathbf{p})$$

The higher ϕ the better the quality of an explainer (beware not to confuse this ϕ with feature attributions from previous definitions). Intuitively, the model's prediction probability should decreases when a feature with positive attribution is removed. Thus, the faithfulness metric could show to what degree that intuition is followed. Intuitively, the method scores explainers for attribution values that have the same impact as the model when excluding/perturbing the feature.
The method has a drawback, as this metric uses correlation it is not defined for small explanations. It is not possible to calculate correlation when the length Θ and $\mathbf{p}$ is 1, as correlation is not defined for a point. And if they consist of two probabilities and feature importances, the correlation is always either 1 or -1. The metric would always assign either the best or worst score to the explanation. Thus, this is not expected behaviour for such a metric. Nonetheless, explanations would often consist of more features, for these explanations the faithfulness metric could still provide insight, by scoring the intuition as described above.

4 Experimental setup

In this section we will discuss simulated experiments for testing the quality of model-agnostic explainers. Two of the experiments are based on those presented in a paper by Ribeiro et al. [1]. The code is provided in an online repository[4]. This is the basis for the experiments in the current paper and they will accordingly be named Lime experiments 5.2 and 5.3. Minor adjustments have been made to run this experiment in python 3.7 rather than 2.7. Additionally, a new experiment using generated data is proposed. All explanations are limited to a maximum of K=10 features. Code for the current paper is available at https://github.com/marnixm/lime_experiments.

opmerking brecht: wtf gebeurt er allemaal, leg volgorde uit

These experiments aim to test the quality of explainers in a quantitative manner. Furthermore, a new measurement for explainer quality from Arya et al. [5] is contrasted to the experiment's results.
In the current paper Shap explanations are included in the experiments.

[4]https://github.com/marcotcr/lime-experiments

While Aas et al. suggest using Shap with an extension for dependent features may be beneficial, its explanations are not as simple as Lime and Shap. Its explanations can only be properly interpreted as clusters of dependent features. Hence, this explainer is not included in the experiments.
The greedy and random explainer from Lime only provide inclusion of features, but not feature attribution. Since a feature attribution is crucial to explainers, these explainers are rejected as baselines. Thus, have been excluded from the experiments.
As mentioned in the faithfulness section, the method does not show desirable behaviour for explanations with two or less features. Instances to which this applies, have been omitted from the experiment's results.
Lastly, the Parzen explainer may find explanations of large size, unlike Lime and Shap. Therefore, the Parzen explanations will consist of only the K most important features (as was the choice in the original Lime experiments).

4.1 Data

The data consists of product reviews from Amazon.com. A review is marked as either positive or negative, thus a binary outcome. The data has been used for several studies, initially by Blitzer et al. [6]. The features in this dataset are the words used in the reviews for each domain. For the current study, the datasets on DVD's, books and kitchen products have been used. All three datasets have approximately 20.000 features and 2000 rows of data. The data is split into a train and test set of respectively 1600 and 400 rows.

4.2 Lime experiment 5.2 Are explanations faithful to the model

In this experiment machine learning methods are used that are interpretable by themselves. Namely sparse logistic regression and decision trees. However, these models are only allowed to use a maximum of K features for each row in the test set. Thus, for all instances, a golden set of features is known. In this experiment, the writers aimed to show that their explainer can find the features used by the model.
A comparison is made of the golden set of features for each instance to the explanations. Thus, scores for recall of golden features from the model can be calculated. Precision would also be an interesting metric to consider. However, the models are asked to provide an explanation of 10 features. If the model itself would use less than 10 features, the explainer could never reach a precision of 1.

The sparse logistic regression model uses a l1 penalty, where the penalty parameter is increased until a maximum of 10 features for each row is used. Similarly, the decision tree model is only allowed to use a maximum of 10 features.
It should be noted that while section 5.1 from the Lime paper [1] describes the use of L2 regularization for the linear regression in their experiments, it is actually L1 regularization that was implemented in their repository. Both Shap and Lime are provided with a budget of 15.000 samples.
Lastly, beware that the title of this experiment is not to be confused with the faithfulness metric. The title actually advertises that the explainer should use the same features as the models themselves.

Lime experiment 5.2 improvements

-Not yet implemented, preliminary version of subsection-
The sparse linear regression model, from experiment 5.2 is provided with an increasingly high penalty, until a maximum of 10 features are used for all instances. More precisely, the features used by model (Θ) and the features provided by the explainer (ξ) are for **all** instances: $|\Theta \cap \xi| < 10$. In practise, the sparse linear regression is provided with such a high penalty, that this intersection has an average of 2-3 features. Thus, the explainer is allowed to provide 10 features, whereas on average 2-3 features are used. As a result, high recall numbers of $> 90\%$ are reported.
Instead, it is suggested to find models with an average (rather than max) of 10 used features for each instance. 1) Large Θ are cut-off at 10 features. 2) Small Θ will have the explainer cut-off at $|\Theta|$ features. 2) seems to have a lot of impact on recall%.
Additionally, for both of these models, the importance of each variable is known. Instead of calculating just recall, we suggest to take the ranking of variables into account. If the explainer find the golden features, but also rank them in accordance to model importance, it should be even better. To this end, the NDCG measure is used. ...
ndcg is not perfect either. Example:
feature is 4th in place, but explanation is only 3 long. Then all features after 3 are cut-off.
–If model uses ¿10 features, we cut of at 10. If model uses ¡10 features, we cut of at that length. This is not known to the explainer.
–state average length of $|\Theta \cap \xi|$
–for tree model we use variable_importances_, which is global?
–in order to have $|\Theta \cap \xi = 10|$ for all instances, we would need to create a lot of

seperate models. Which is probably not the point // –ndcg calculated with minimum length of both vectors. To adjust for too small faulty explanation, the ndcg is multiplied with the recall for that instance

4.3 Lime experiment 5.3 Should I trust this prediction

Trustworthiness of the explainers is assessed by considering the impact of features on the model and the explanation. In this experiment it is assumed that the user can identify a number of features that are not trustworthy and that the user would not want to include in the model. For each instance in the dataset, 25% of the features are sampled to be untrustworthy. We then compare how the predictions of the model and explainer change by removing the effect of the untrustworthy features. In other words, the user 'discounts' the effect of untrustworthy features.

Discounting for the model is done by replacing untrustworthy feature values with the background value. The adjusted instance is passed through the classifier for a new predicted probability. In case of the explainer, the initial prediction is formed by the (local) mean prediction added to all explained features attributions. Discounting for the explainer is the initial prediction subtracted by the explained model impact of each feature that is untrustworthy in that instance.

For both the model and explainer, the experiment will compare the initial prediction with the prediction after discounting untrustworthy features. If the untrustworthy features change the prediction of the model, the explainer should show the same behaviour. It is pointed out that the classification problem is binary, thus a change of prediction is defined by going from negative to positive classification, or vise versa.

Considering the change of prediction results in a vector of trusted and mistrusted instances. Precision and recall can be calculated from these vectors. In the original paper, the F1 score was reported for two datasets.

Trustworthiness can be tested for any classification model. For this experiment four models are considered: sparse logistic regression (LR), decision tree (Tree), support vector machine (SVM) and random forest (RF). Model parameters are as in the Lime paper. Only the solver for the logistic regression was changed to 'lbfgs', since the old solver was no longer supported.

Lime experiment 5.3 improvements

-Not yet implemented, preliminary version of subsection- In the original paper, it was decided to compare whether both the model and explainer changed predictions due to the removal of untrustworthy features. However, the model and explainer should in the first place, have the same prediction. While this is always the case for Shap values, Lime and Parzen explanations may initially be wrong. This had not been taken into account in Robeiro et al. [1]. To adjust for this effect, a new 'accuracy' score is introduced. The accuracy of the explainer is decreased if the prediction, before removal of untrustworthy features is, not the same as the model. We note again that the model has a binary outcome, siding with positive and negative at the threshold of 0.5.
–according to local accuracy property
In the improved experiment, the f1 score is multiplied with the accuracy score (it receives a penalty) and will be named f1 adjusted.

4.4 Experiments with generated data

To be added
–Use lime tabular package, standard lime kernel
–Shap uses Kmean, otherwise too slow
–parzen windows parameters (sigmas)???

5 Results

This section will present the results of all simulated user experiments described in the previous section.

5.1 Experiment 5.2

In this experiment small interpretable models are trained such that a golden standard of features is known. It should be noted that the average amount of golden features, is only between 2-3. Whereas the explainers are given a budget of 10 features for their explanations. Explanations are generated for each instance in the test dataset. The average recall is shown in figure 3. Even though the explanation methods are model-agnostic, the figure shows varying results. Shap has a perfect recall for the logistic regression model. Interestingly, for the decision tree model Shap's performance is lower. Shap

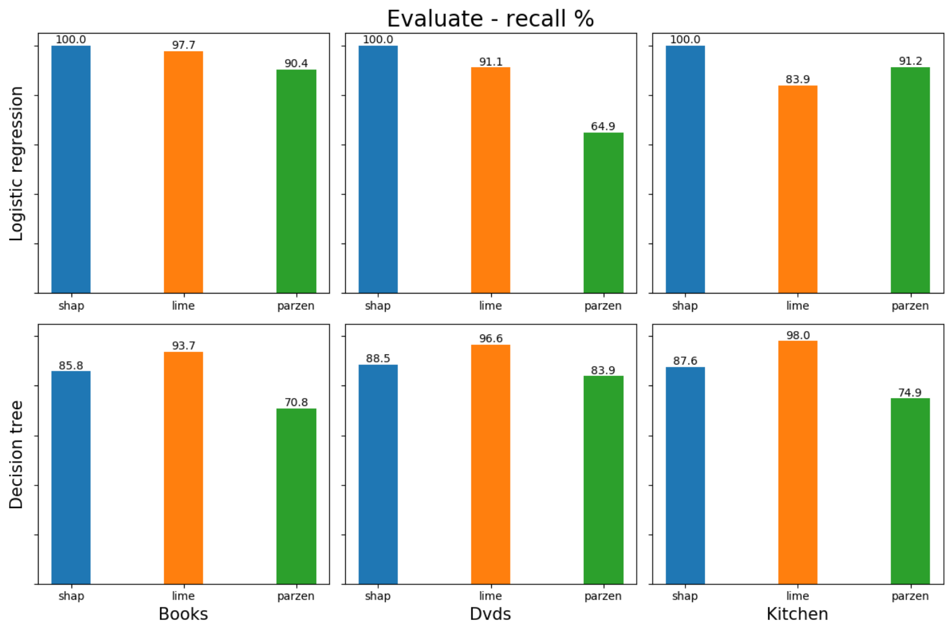


Figure 3: Experiment 5.2 recall %

is able to get a recall of approximately 85-89% recall. As presented in the Lime paper, the method is able to get a recall of over 90% for most datasets and models. With an exception for logistic regression with the Kitchen dataset, where it scores a 83.9%. In the original paper, that particular dataset was not included in the results section. Nonetheless, Lime outperforms both other explainers on the decision tree model. The Parzen explainer is used as baseline. Accordingly, it shows the lowest performance. A single exception is it outperforming Lime on the Kitchen dataset with logistic regression, though it is still worse than Shap.
In summary, Shap has superior performance on the logistic regression model. However, for the decision tree model, its recall scores are lower than Lime for all datasets. The baseline reaches lowest scores, with a single exception over Lime.
According to these results, an argument is put forward that Shap's assumption of independent features is punished by models which allow that dependency. This speculation would agree with given results.

Faithfulness metric

The average faithfulness scores for this experiment are shown in figure 4. Note that the higher the faithfulness score ϕ, the better the explainer should be. The figure demonstrates that Shap has dominantly higher faithfulness

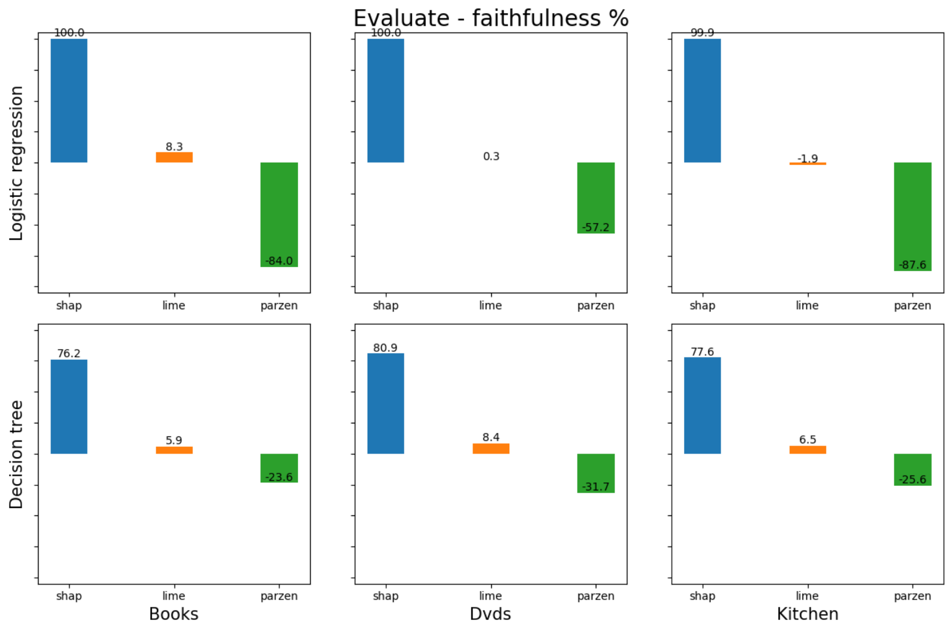


Figure 4: Experiment 5.2 faithfulness metric %

scores than the other explainers on all datasets. In second place comes Lime with scores fluctuating from approximately -2% to 8%. Lastly, the baseline has the lowest scores. For all datasets and models the faithfulness score is negative.
However, upon further investigation the faithfulness measure seems impacted by small feature attributions. Since the explainers are asked to find an explanation of size K=10, they will assign some features a small attribution. Even if, for that instance, less than 10 features are used. Remind that the experiment resulted in an average of 2-3 features per instance.
Scores where the faithfulness is nearly one, seem to come from above described explanations. The model's prediction does not change, because the features is not included. The explainer however, has assigned a tiny attribution. A consequence seems to be that these explanations are assigned scores of approximately 1 and -1. Especially for Lime, the faithfulness scores fluc-

tuate between these extremes. As a result, the average faithfulness is close to 0.
It is argued that this behaviour of the faithfulness metric is not desirable, having in mind that the explainers are forced to a K-sized explanation.

5.2 Experiment 5.2 improved

–Linear regression model can take length of 50+ features. As a result, the average recall is low. The explainer can only provide 10 features. Maximize the division by 10.

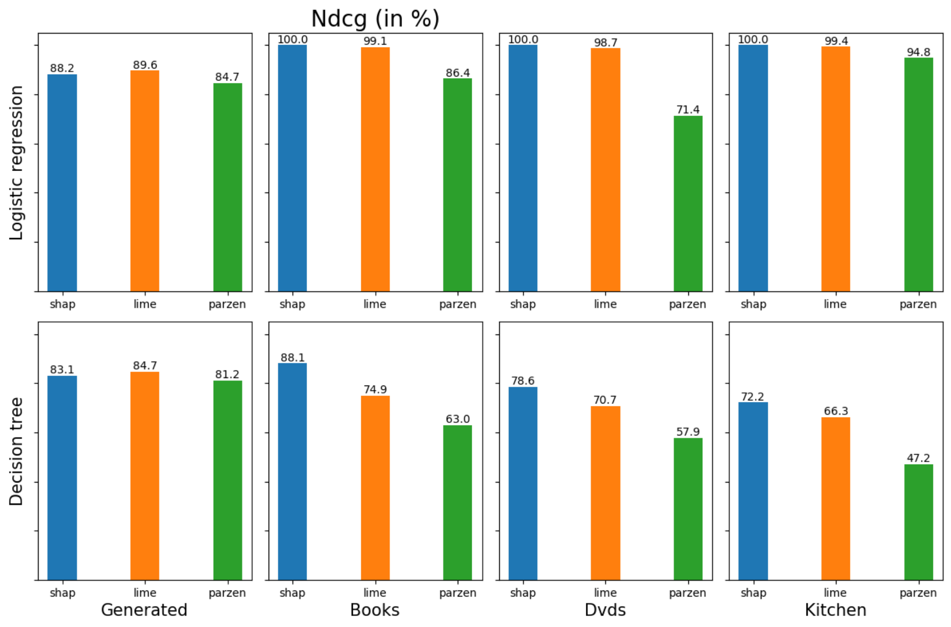


Figure 5: Experiment 5.2 ndcg metric %

5.3 Experiment 5.3

For each instance of the data an arbitrarily selected 25% of features is deemed untrustworthy. The prediction for the model and explainer are compared before and after the untrustworthy features are discounted. The adjusted f1 score is reported in table 1.
For all three datasets, Shap is able to produce the highest adjusted f1 score for LR, SVM and RF models. However, Lime achieves the best score for the

Toevoegen van t-test scores van Lime/Shap?

decision tree model, similarly to experiment 5.2.
Individual scores for recall, precision and accuracy measure are included in appendix section 7.0.2. Shap produces a perfect score on accuracy. That is due to the local accuracy property, which Shapley values are guaranteed to satisfy. The accuracy score do give a penalty to the Lime and Parzen method, as their initial classification differs from that of the predictive model.

Dit stuk over resultaten ook verplaatsen naar appendix?

Parzen is most heavily punished by its accuracy scores. The results show that Parzen often wrongly classifies the initial prediction. As a result, for this experiment Parzen does not compete with either Lime or Shap.

	Books				DVDs				Kitchen			
	LR	Tree	SVM	RF	LR	Tree	SVM	RF	LR	Tree	SVM	RF
Shap	**96.8**	93.9	**96.7**	**98.5**	**96.5**	94.8	**95.9**	**98.5**	**97.8**	95.0	**97.6**	**99.2**
Lime	95.8	**97.3**	94.9	93.9	95.5	**97.7**	95.1	97.4	97.4	**97.6**	97.5	98.3
Parzen	53.7	54.3	65.6	59.6	53.5	48.7	55.5	47.8	34.8	58.5	48.0	64.5

Table 1: Experiment 5.3 - adjusted f1 score %

Faithfulness metric

To be added

5.4 Results summary

To be added
–Overall, Shap seems to be most reliable explainer. As follows from the theory
–Interestingly, Lime may be better for decision trees specifically.Raises the question if that is the case for other models too.
–Lime seems to get close to Shap results, viable faster because it is faster.

6 Discussion & Future work

– Linear post hoc model-agnostic explainers for highly non-linear models
– Vervolg met: Herhaal gevaren van gebruik van post hoc explainers
– Een methode om het aantal variabelen voor een explainer te kiezen. Soort

van statistische test om te bepalen wat de optimale K is (voor lime/shap)
– Decision tree as explanation model
– Kwaliteit van explainers meten met een betere metric?
– Faithfulness niet meegenomen indien vector ¡=2

References

[1] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should i trust you?" Explaining the predictions of any classifier.

[2] Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions.

[3] Aas, K., Jullum, M., & Løland, A. (2019). Explaining individual predictions when features are dependent: More accurate approximations to Shapley values.

[4] Camburu, O. M., Giunchiglia, E., Foerster, J., Lukasiewicz, T., & Blunsom, P. (2019). Can I Trust the Explainer? Verifying Post-hoc Explanatory Methods.

[5] Arya, V., Bellamy, R. K. E., Chen, P.-Y., Dhurandhar, A., Hind, M., Hoffman, S. C., Houde, S., Liao, Q., Luss, R., Mojsilovic, A., Mourad, S., Pedemonte, P., Raghavendra, R., Richards, J., Sattigeri, P., Shanmugam, K., Singh, M., Varschney, K., Wei, D. & Zhang, Y. (2019). One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques.

[6] Blitzer, J., Dredze, M., Pereira, F. (2007). Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification.

[7] Slack, D., Hilgard, S., Jia, E., Singh, S., & Lakkaraju, H. (2019). How can we fool LIME and SHAP? Adversarial Attacks on Post hoc Explanation Methods.

[8] Lundberg, S. M., Erion, G. G., & Lee, S.-I. (2018). Consistent Individualized Feature Attribution for Tree Ensembles.

[9] Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., & Müller, K. R. (2010). How to explain individual classification decisions.

[10] Shapley, L. S. (1953). A value for n-person games.

7 Appendix

To be revised

Lime

Lime finds the explanation that is the most locally faithful, while still being interpretable. For a linear explanation, local faithfulness $L(f, g, \pi_x)$ is formally defined by:

$$L(f, g, \pi_x) = \sum_{z,z' \subseteq Z} \pi_x(z)\ (f(z) - g(z'))^2$$

Where z and z' are perturbed samples.
The (heuristically chosen) kernel is defined as:

$$\pi_x(z) = exp\left(\frac{-D(x,z)^2}{\sigma^2}\right)$$

With D is some distance function with width σ. The distance function for text classification is cosine distance.

Shap

Subsection will include any method/formulas/definitions that would be too much for method section

7.0.1 Desirable properties

Desirable properties for additive feature attribution methods.
-to be completed-

Property 1: Local accuracy

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i$$

Property 2: Missingness

$$x'_i = 0 \Longrightarrow \phi_i = 0$$

Property 3: Consistency
If $f'_x(z') - f'_x(z' \backslash i) \geq f_x(z') - f_x(z' \backslash i)$ for all inputs $z' \in \{0,1\}^M$, then

$\phi_i(f', x) \geq \phi_i(f, x)$
A unique additive feature explanation model follows from these properties:

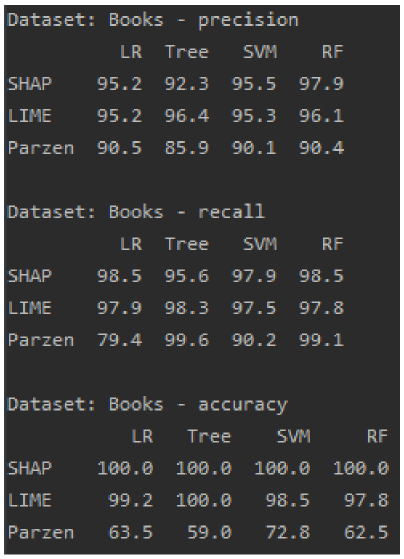
$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!\,(M - |z'| - 1)!}{M!}[f_x(z') - f_x(z' \setminus i)]$$

7.0.2 Shap value estimation

how Shap estimates Shapley values with higher sample efficiency [3]

Additional results

Additional results that would crowd the results section.
Precision, recall and accuracy scores for Lime experiment 5.3:

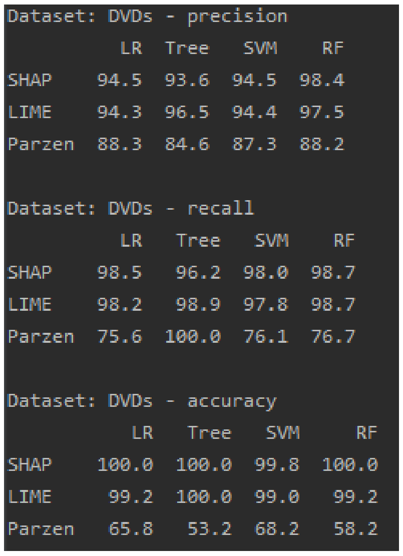


```
Dataset: Books - precision
          LR  Tree   SVM    RF
SHAP    95.2  92.3  95.5  97.9
LIME    95.2  96.4  95.3  96.1
Parzen  90.5  85.9  90.1  90.4

Dataset: Books - recall
          LR  Tree   SVM    RF
SHAP    98.5  95.6  97.9  98.5
LIME    97.9  98.3  97.5  97.8
Parzen  79.4  99.6  90.2  99.1

Dataset: Books - accuracy
           LR   Tree    SVM     RF
SHAP    100.0  100.0  100.0  100.0
LIME     99.2  100.0   98.5   97.8
Parzen   63.5   59.0   72.8   62.5
```

Figure 6: Experiment 5.3 Books%

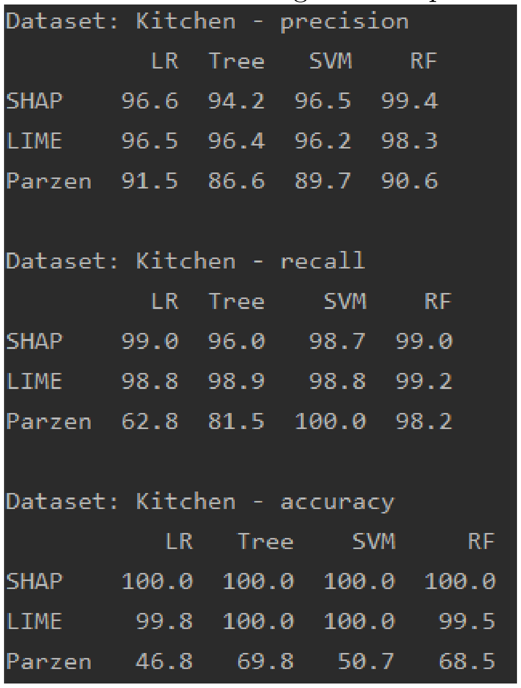
Code

–may include–
Shortly describe the code



```
Dataset: DVDs - precision
          LR  Tree   SVM    RF
SHAP    94.5  93.6  94.5  98.4
LIME    94.3  96.5  94.4  97.5
Parzen  88.3  84.6  87.3  88.2

Dataset: DVDs - recall
          LR   Tree   SVM    RF
SHAP    98.5   96.2  98.0  98.7
LIME    98.2   98.9  97.8  98.7
Parzen  75.6  100.0  76.1  76.7

Dataset: DVDs - accuracy
           LR   Tree   SVM     RF
SHAP    100.0  100.0  99.8  100.0
LIME     99.2  100.0  99.0   99.2
Parzen   65.8   53.2  68.2   58.2
```

Figure 7: Experiment 5.3 DVDs%



```
Dataset: Kitchen - precision
          LR  Tree   SVM    RF
SHAP    96.6  94.2  96.5  99.4
LIME    96.5  96.4  96.2  98.3
Parzen  91.5  86.6  89.7  90.6

Dataset: Kitchen - recall
          LR  Tree    SVM    RF
SHAP    99.0  96.0   98.7  99.0
LIME    98.8  98.9   98.8  99.2
Parzen  62.8  81.5  100.0  98.2

Dataset: Kitchen - accuracy
           LR   Tree    SVM     RF
SHAP    100.0  100.0  100.0  100.0
LIME     99.8  100.0  100.0   99.5
Parzen   46.8   69.8   50.7   68.5
```

Figure 8: Experiment 5.3 Kitchen%